

作业提交服务器：ftp://192.168.134.123

用户名：uploader

密码：sdt7%5252@3

！ 注意事项

1. 每题需要提交一个cpp文件，确保cpp文件可以正确编译。cpp文件用题目编号命名，除非有特别说明。
2. 每题需要提交至少一张运行结果截图。将所有题目的截图放在一个pdf文件中。
3. 将所有cpp文件和一个pdf文件打包成zip文件，用自己的学号命名，并上传至ftp指定文件夹。

举例说明：比如学号是1001，本次有3个题目，那么最终提交1个1001.zip文件，其中包含3个cpp文件（分别是1.cpp, 2.cpp, 3.cpp）和1个pdf文件，其中pdf文件中包含至少3个运行结果的截图。第1次作业上传至hm1文件夹。

第2次作业

提交时间：2024年4月21日22:00

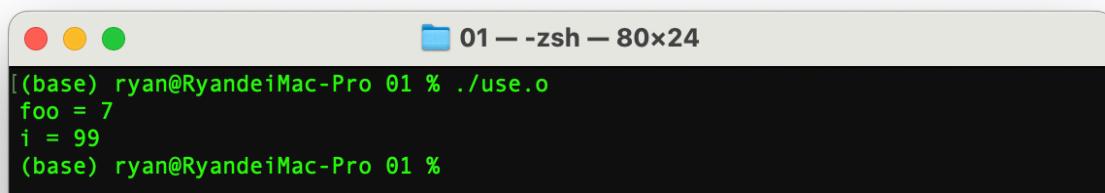
1. 创建3个文件my.h、my.cpp和use.cpp。头文件my.h包含：

```
1 extern int foo;
2 void print_foo();
3 void print(int);
```

源文件my.cpp包含my.h和iostream，定义print_foo()，此函数用cout来打印foo的值，my.cpp中还应定义函数print(int i)，用cout打印i的值。

源文件use.cpp包含my.h，定义主函数main()，主函数中将foo的值置为7，用print_foo()打印它，并用print()打印整型值99。注意use.cpp并不包含iostream，因为它并不直接使用标准库中的功能。

编译这些文件，运行生成的可执行文件（比如use.o文件），报告运行结果。



```
01 — -zsh — 80x24
(base) ryan@RyandeiMac-Pro 01 % ./use.o
foo = 7
i = 99
(base) ryan@RyandeiMac-Pro 01 %
```

2. 编写三个函数swap_v(int, int)、swap_r(int &, int &)、swap_cr(const int &, const int &)。每个函数有如下函数体：

```
1 { int temp; temp = a; a = b; b = temp; }
```

其中a和b是参数名。尝试用如下代码调用这三个函数：

```
1 int x = 7, y = 9;
2 swap_v(x, y);
```

```

3      swap_r(x, y);
4      swap_cr(x, y);
5      swap_v(7, 9);
6      swap_r(7, 9);
7      swap_cr(7, 9);
8
9      const int cx = 7, cy = 9;
10     swap_v(cx, cy);
11     swap_r(cx, cy);
12     swap_cr(cx, cy);
13     swap_v(7.7, 9.9);
14     swap_r(7.7, 9.9);
15     swap_cr(7.7, 9.9);
16
17     double dx = 7.7, dy = 9.9;
18     swap_v(dx, dy);
19     swap_r(dx, dy);
20     swap_cr(dx, dy);
21     swap_v(7.7, 9.9);
22     swap_r(7.7, 9.9);
23     swap_cr(7.7, 9.9);

```

指出所有的编译错误，解释为什么会有这样的错误。对每个编译通过的swap调用，参数值是否真正的被交换了？解释原因。（本题不需要提交cpp文件，只需要提交pdf文件）

3. 编写一个程序，由一个文件组成，其中包含三个名空间X、Y和Z，使得如下main函数能正确运行：

```

1  int main()
2  {
3      X::var = 7;
4      X::print(); //打印名空间x中的var
5      using namespace Y;
6      var = 9;
7      print(); //打印名空间y中的var
8      {
9          using Z::var;
10         using Z::print;
11         var = 11;
12         print(); //打印名空间z中的var
13     }
14     print(); //打印名空间y中的var
15     X::print(); //打印名空间x中的var
16 }

```

每个名空间中需定义一个名为var的变量和一个名为print的函数，该函数用cout输出对应的var值。

```
03 — -zsh — 80x24
(base) ryan@RyandeiMac-Pro 03 % ./03.o
7
9
11
9
7
(base) ryan@RyandeiMac-Pro 03 %
```

4. 编写一个猜谜游戏“bulls and cows”。程序用vector存储四个不相同的0~9之间的数字，比如1234。用户的任务是通过重复猜测来找到这四个数字。比如存储的数是1234而用户猜1359，那么程序输出“1 bull and 1 cow”，因为用户的猜测中有一个数字是正确的且在正确的位置上（数字1，所以输出1 bull），有一个数字是正确的但在错误的位置上（数字3，所以输出1 cow）。反复这一猜测过程直到用户找到这四个数字，即四个数字都正确且都在正确的位置上。下面给出了main函数：

```
1  int main()
2  {
3      std::vector<int> secret = get_secret_number(); //随机生成4个不重复的0~9之间的数字
4      std::pair<int, int> result; //first存放bull的数量, second存放cow的数量
5
6      do {
7          std::cout << "Enter your guess (4 unique digits): ";
8          std::vector<int> guess = get_user_guess(); //获取用户猜测
9          result = check_guess(secret, guess); //计算用户猜测的正确性
10         std::cout << result.first << " bull(s) and " << result.second << "
cow(s)" << std::endl;
11     } while (result.first < 4); //bull < 4 : 没有完全猜对
12
13     std::cout << "Congratulations! You've guessed the number." << std::endl;
14     return 0;
15 }
```

请在此基础上完成函数get_secret_number()、get_user_guess()和check_guess()。你需要设计合适的函数原型并给出正确的函数实现，使得整个程序可以正确运行。提示：1) std::pair类型参考教材379~381页第11.2.3节；2) 生成随机数参考教材659~663页第17.4节。一种可能的运行结果如下图所示：

```
04 — -zsh — 102x24
(base) ryan@ryandeMac-Studio 04 % ./04.o
Enter your guess (4 unique digits): 0 1 2 3
1 bull(s) and 0 cow(s)
Enter your guess (4 unique digits): 4 5 6 7
0 bull(s) and 2 cow(s)
Enter your guess (4 unique digits): 8 9 0 1
1 bull(s) and 1 cow(s)
Enter your guess (4 unique digits): 0 9 4 5
2 bull(s) and 1 cow(s)
Enter your guess (4 unique digits): 0 9 5 6
4 bull(s) and 0 cow(s)
Congratulations! You've guessed the number.
(base) ryan@ryandeMac-Studio 04 %
```

5. 阿拉伯数字的中文读法遵循如下规则：
- 1、四位以内的数，按照数位顺序，从高位读起。
 - 2、四位以上的数，先从右向左四位分级，然后从最高级起，依次读亿级、万级、个级。读出各级里的数和它们的级名。亿级里的数，按照个级的数的读法来读，再在后面加上一个“亿”字；万级里的数，按照个级的数的

读法来读，再在后面加上一个“万”字；

3、每级末尾不管有几个“0”，都不读；其他数位上有一个“0”或连续几个“0”，都只读一个零。

下面看一个具体的例子。比如200001010020，根据四位分级，可以分成3级：2000 0101 0020。第一级的级别是亿，零都在末尾，所以读成“二十亿”。第二级的级别是万，两个0不处于末尾，所以读成“零一百零一万”。第三级的级别是个，两个连续的0只读1个零，所以读成“零二十”。所以，200001010020读成“二十亿零一百零一万零二十”。

编写一个程序，首先提示用户本地机器上int和long long类型所能表示的最大整数，然后提示用户输入一个不超过long long类型所能表示的最大整数，并输出该整数的中文读法。

```
05 -- -zsh -- 80x24
(base) ryan@RyandeiMac-Pro 05 % ./05.o
int能表示的最大整数是：2147483647
long long能表示的最大整数是：9223372036854775807
请输入一个合适大小的整数：
2147483647
二十一亿四千七百四十八万三千六百四十七
(base) ryan@RyandeiMac-Pro 05 % ./05.o
int能表示的最大整数是：2147483647
long long能表示的最大整数是：9223372036854775807
请输入一个合适大小的整数：
9223372036854775807
九百二十二亿亿三千三百七十二万亿零三百六十八亿五千四百七十七万五千八百零七
(base) ryan@RyandeiMac-Pro 05 %
```

更多的测试用例

1	0 零
2	1 一
3	2 二
4	3 三
5	4 四
6	5 五
7	6 六
8	7 七
9	8 八
10	9 九
11	10 一十
12	11 一十一
13	110 一百一十
14	111 一百一十一
15	100 一百
16	102 一百零二
17	1020 一千零二十
18	1001 一千零一
19	1015 一千零一十五
20	1000 一千
21	10000 一万
22	20010 二万零一十
23	20001 二万零一
24	100000 一十万
25	1000000 一百万
26	10000000 一千万
27	100000000 一亿

28	1000000000	一十亿
29	1000001000	一十亿一千
30	1000000100	一十亿零一百
31	200010	二十万零一十
32	2000105	二百万零一百零五
33	20001007	二千万一千零七
34	2000100190	二十亿零一十万零一百九十
35	1040010000	一十亿四千零一万
36	200012301	二亿零一万二千三百零一
37	2005010010	二十亿零五百零一万零一十
38	4009060200	四十亿零九百零六万零二百
39	4294967295	四十二亿九千四百九十六万七千二百九十五